第4章

XC3S250E活用チュートリアル

FPGA開発ツールの使い方

池澤陽一

ここでは，付属FPGA基板の活用に不可欠なFPGA/PLD開発ツール「ISE WebPACK 9.1i」の使い方を解説する．XC3S250E向けの簡単な回路を設計し，FPGAを動作させる．実際に行う手順を具体的に示すので，同じように操作すればFPGA設計を体験できる（本誌付属DVD-ROMから開発ツールをインストールする手順については，pp.60-65のAppendixを参照）．（編集部）

ここでは，ISE WebPACK 9.1iを使用して，プロジェクト作成からFPGAへダウンロードする手順を説明します．ISEはXilinx社のFPGA/PLD開発ツールです．もともとはIntegrated Software Environmentの略称であり，統合設計環境を意味しています．設計入力，論理合成，配置配線，タイミング解析，FPGAへの回路データのダウンロードなど，FPGA開発の一連の作業を行うことができます．

ISE WebPACK 9.1iは，本誌付属DVD-ROMに収録しています．定期的にバージョンアップされていますが，最近は基本的な操作方法はほとんど変わっていないので，今後のバージョンでもある程度対応できると思います．

1．付属FPGA基板とサンプル回路の準備

サンプル回路は，LEDを点滅させる回路とします．回路の構成を**図1**に示します．

回路は，top（**リスト1**）を最上位ブロックとして，三つの機能ブロックで構成しています．

smallcntrブロック（**リスト2**）の4ビット・カウンタの値に従って，decodeブロック（**リスト3**）で生成したパターンを出力し，LEDを点滅させます．

付属FPGA基板には，33MHzのクロック発振器が実装されているものとします．このクロックをそのまま使用すると，LEDの点滅動作を目で見ることができません．そこでtimブロック（**リスト4**）では，周期の長いイネーブル信号を作っています．smallcntrは，このイネーブル信号が“H”の時のみカウントアップするように動作します．

reset信号は，“L”アクティブです．resetが“L”の時，LEDは消灯し続けます．resetが“H”になると，LEDが点滅を開始します．reset信号を発生させるために，付属FPGA基板にはスイッチを実装しておいてください．

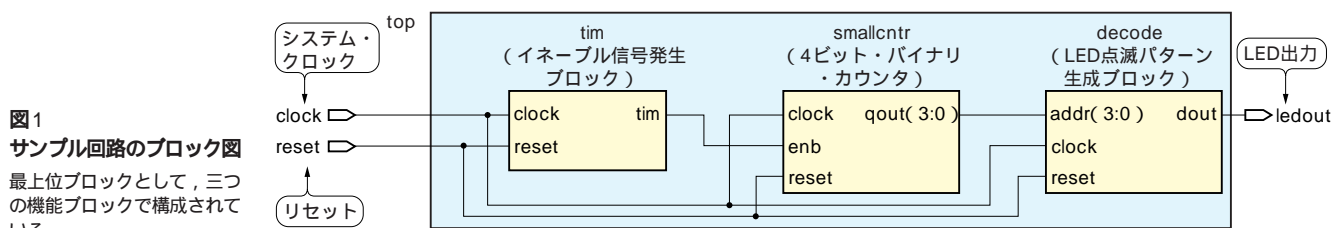



図1 サンプル回路のブロック図
最上位ブロックとして，三つの機能ブロックで構成されている．

KeyWord
FPGA，ISE WebPACK，プロジェクト，論理合成，ピン配置，配置配線，プログラム，ダウンロード，コンフィグレーション

リスト1～リスト4の設計データは，ブロック名と同じファイル名（拡張子.vhd）で，今回はC:\vhdldemoというフォルダを作成してその中に用意しておいてください．

2. ISE WebPACKによるFPGA開発を体験する

ISE WebPACKによるFPGA開発フローを図2に示します．

リスト1　最上位ブロックtopのVHDLソース・コード（top.vhd）

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity top is
      port (clock : in std_logic;
            reset : in std_logic;
            ledout : out std_logic);
end top;

architecture arc_top of top is

      signal cntout : std_logic_vector ( 3 downto 0);
      signal enable : std_logic;

      component tim
      port(clock : in std_logic;
           reset : in std_logic;
           tim   : out std_logic);
      end component;

      component smallcntr
      port (clock : in std_logic;
            reset : in std_logic;
            enb   : in std_logic;
            qout  : out std_logic_vector(3 downto 0));
      end component;

      component decode
      port (addr: in std_logic_vector(3 downto 0);
           clock: in std_logic;
           reset: in std_logic;
            dout: out std_logic);
      end component;

begin

       u1: tim port map(
                 clock => clock,
                 reset => reset,
                   tim => enable);

       u2: smallcntr port map(
                 clock => clock,
                 reset =>  reset,
                   enb => enable,
                  qout => cntout);

       u3 : decode port map (
                 addr   => cntout,
                 clock => clock,
                 reset => reset,
                  dout => ledout);
end arc_top;
```

● ISE WebPACKの起動

Windowsのデスクトップ上にあるXilinx ISE 9.1iのアイコンをダブル・クリックするか，スタートから「プログラム」→「Xilinx ISE 9.1i」→「Project Navigator」を選択してISE WebPACKを起動します．すると，Project Navigatorの画面が開きます（図3）．

● プロジェクトを新しく作成する

FPGA開発は，プロジェクトの新規作成から始めます．プロジェクトとは，開発ツールが設計データを扱う一つの

リスト2　smallcntrブロックのVHDLソース・コード（smallcntr.vhd）

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity smallcntr is
      port (clock : in std_logic;
            reset : in std_logic;
            enb   : in std_logic;
             qout : out std_logic_vector(3 downto 0));
end smallcntr;

architecture arc_cnt of smallcntr is

      signal temp : std_logic_vector (3 downto 0);

begin

      process (clock,reset)
      begin
              if reset='0' then
                     temp <= "0000";
              elsif  clock'event and clock='1' then
                  if  enb = '1' then
                     temp <= temp + 1;
                  else
                     temp <= temp;
                  end if;
              end if;
      end process;

      qout <= temp;

end arc_cnt;
```

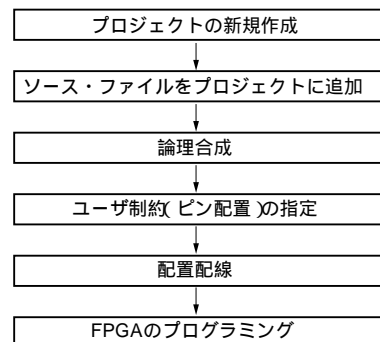



図2　ISE WebPACKによるFPGA開発フロー
本章で解説する手順を示している．

リスト3　decode ブロックのVHDL ソース・コード(decode.vhd)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;


entity decode is
    Port ( addr : in std_logic_vector(3 downto 0);
           clock: in std_logic;
           reset: in std_logic;
           dout : out std_logic);
end decode;

architecture arc_dec of decode is

begin

    process (clock,reset)
    begin
        if reset = '0' then
            dout <= '0';

        elsif clock'event and clock = '1' then

           case addr is

                when "0000" => dout <= '1';
                when "0001" => dout <= '0';
                when "0010" => dout <= '0';
                when "0011" => dout <= '0';
                when "0100" => dout <= '0';
                when "0101" => dout <= '1';
                when "0110" => dout <= '0';
                when "0111" => dout <= '0';
                when "1000" => dout <= '0';
                when "1001" => dout <= '0';
                when "1010" => dout <= '1';
                when "1011" => dout <= '0';
                when "1100" => dout <= '0';
                when "1101" => dout <= '0';
                when "1110" => dout <= '0';
                when "1111" => dout <= '0';
                when others => null;

           end case;
       end if;
   end process;
end arc_dec;
```

リスト4　tim ブロックのVHDL ソース・コード(tim.vhd)

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity tim is
    port (clock,reset : in std_logic;
                  tim : out std_logic );
end tim;

architecture  tim_arc of tim is

    signal dat_q : std_logic_vector(19 downto 0);

begin
    process (clock,reset)
    begin
          if reset = '0' then
             dat_q <= ( others=>'0');
          elsif clock'event and clock='1' then
             dat_q <= dat_q + 1;
          end if;
    end process;


    process (clock,reset)
    begin
          if reset = '0' then
                    tim <= '0';
          elsif clock'event and clock='1' then
             if dat_q = X"0000F" then
                    tim <= '1';
             else
                    tim <= '0';

             end if;
          end if;
     end process;

end tim_arc;
```

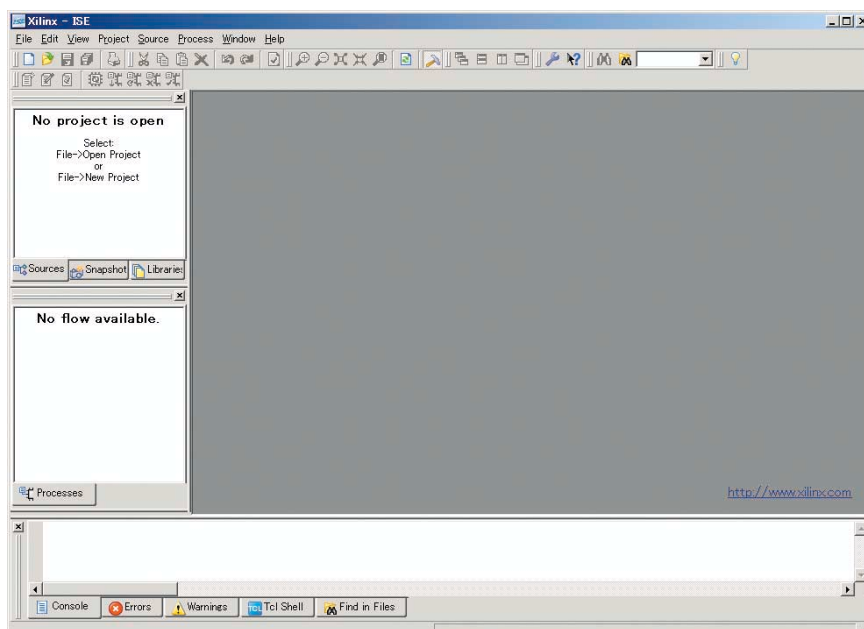
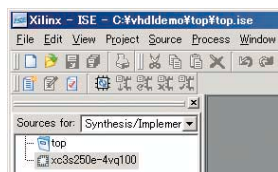
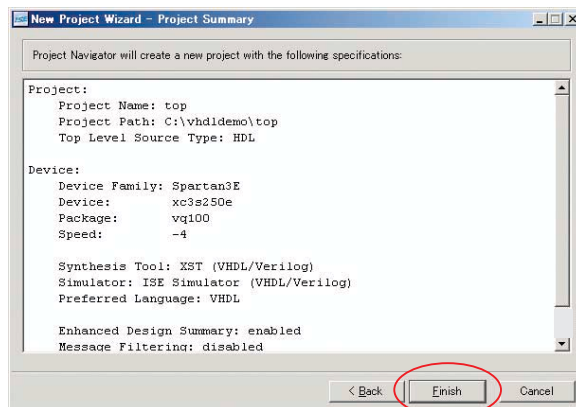
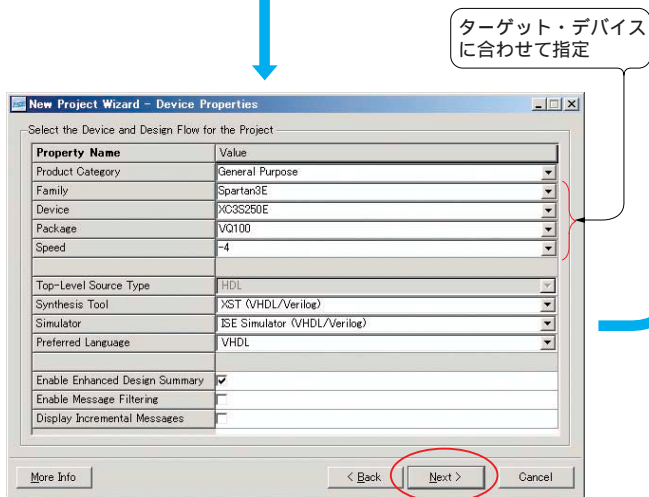
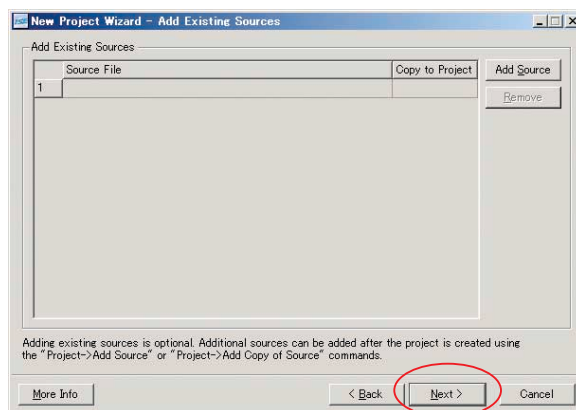
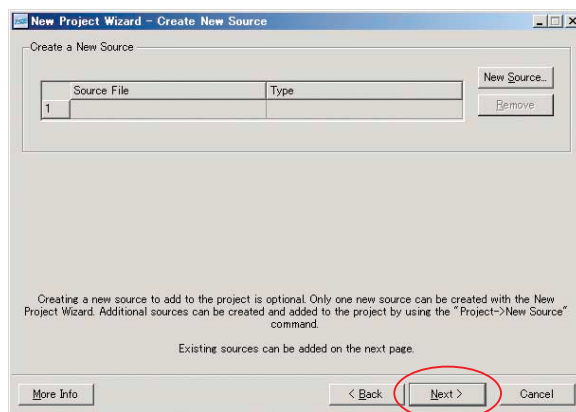
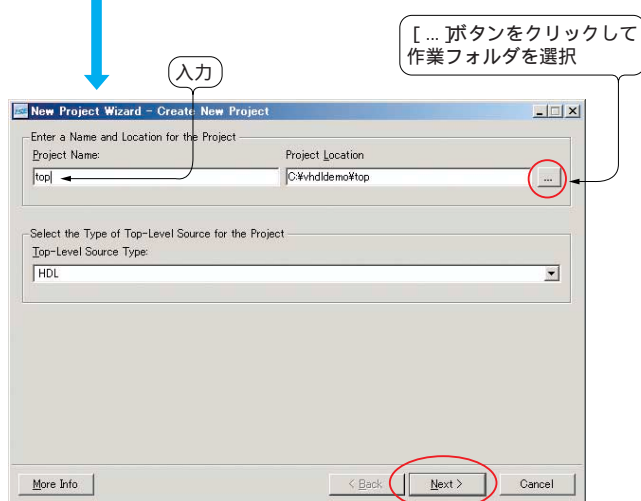
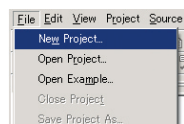



図3
Project Navigator
ISE WebPACKの起動画面である.

単位です．開発ツール自身が使う作業フォルダと考えればイメージしやすいと思います．

Project Navigatorのメニューから「File」→「New Project」を選択すると，New Project Wizardが開きます（図4）．

1ページ目（Create New Project）では，まず，Project Location欄の横にある［...］ボタンをクリックして，「C:\vhdldemo」を選択します．次に，Project Name欄に「top」

図4
プロジェクトの新規作成

と入力します．このとき，Project Location欄にも「top」がサブフォルダとして追加されます．Top-Level Source Type欄が「HDL」となっていることを確認して，［Next］ボタンをクリックします．

2ページ目（Device Properties）では，ターゲットFPGAの指定を行います．付属FPGA基板には，XC3S250E-VQ100-4が搭載されています．そこで，Family欄を「Spartan3E」に，Device欄を「XC3S250E」，Package欄を「VQ100」，Speed欄を「−4」に上から順番に設定します．それ以外の欄は，デフォルトのままでかまいません．設定が終わったら，［Next］ボタンをクリックします．

3ページ目（Create New Source）と4ページ目（Add Existing Sources）は，設計データを作成，読み込む際の指定です．ここではプロジェクトの新規作成だけを行うので，何もせず［Next］ボタンをクリックします．

5ページ目（Project Summary）では，設定した内容が表示されます．内容を確認して［Finish］ボタンをクリックします．

プロジェクトの新規作成が終わると，Project NavigatorのSources forウィンドウに，プロジェクト名とターゲットFPGAが表示されます．

● 設計データをプロジェクトに読み込む

あらかじめ用意している設計データをプロジェクトに読

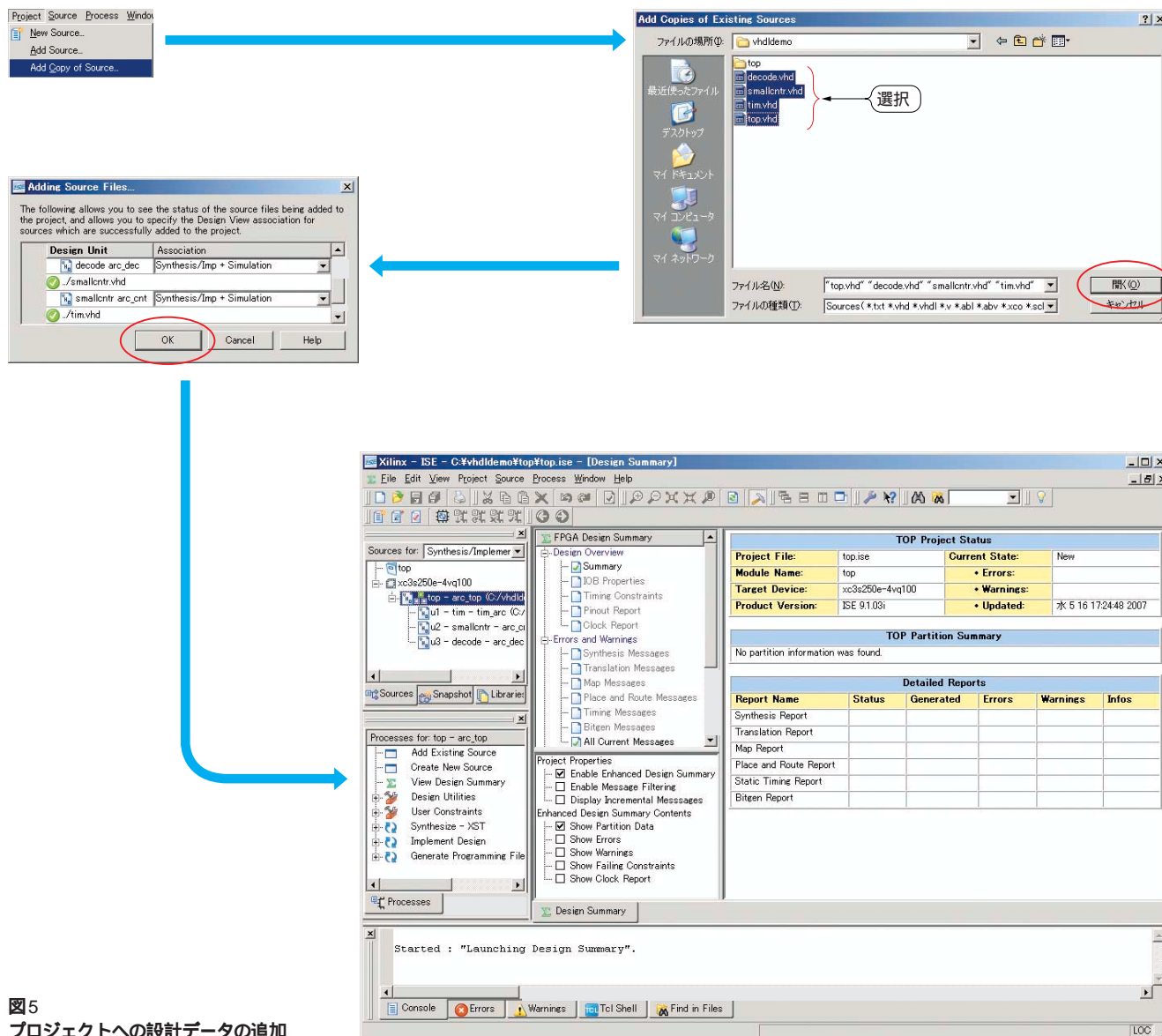



図5
プロジェクトへの設計データの追加

み込みます(**図5**).

Project Navigatorのメニューから「Project」→「Add Copy of Sources」を選択します.

Add Copy of Existing Sourcesウィンドウが開くので,設計データを用意してあるC:\vhdldemoフォルダを参照して,4個のVHDLソース・ファイルを選択して[開く(O)]ボタンをクリックします.

Adding Source Files...ウィンドウで,指定したファイルが読み込まれたこと,Association欄が「Synthesis/Imp + Simulation」になっていることを確認して[OK]ボタンをクリックします.

Project NavigatorのSources forウィンドウには,最初は最上位ブロックのtopだけが表示されます.[+]でツリーを展開すると,サブブロックも表示されます.Project Navigatorはソース・ファイルの階層を自動的に解析します.

● 論理合成を行う

VHDLやVerilog HDLで記述したコードをゲート・レベルの回路に変換することを,論理合成と言います.論理合成は,ISE WebPACKが持つXST(Xilinx Synthesis Technology)という機能を使います.

Project NavigatorのSources forウィンドウで最上位ブロックのソース・ファイル(top.vhd)を選択し,Processes forウィンドウの「Synthesize-XST」をダブル・クリックします(**図6**).Sources forウィンドウで選択したファイルは最上位ブロックとして処理され,その下の階層にあるすべてのソース・コードに対して論理合成を実行します.

論理合成が正常に終了したら,Synthesize-XSTの前に緑のチェックが付きます.

● ピン配置の指定

FPGAは,I/Oピンの配置をある程度自由に決めることができるLSIです.しかし付属FPGA基板では,クロック発振器からのクロック信号やLED出力信号は,基板上の配線により,つなぐべきピンが決まっています.

ピン配置は,FPGA開発ツールにユーザ制約を与えることによって行います.ユーザ制約を行うためのucfファイルを作成し,ISEのエディタ機能を使って指定するという手順になります(**図7**).

(1)ユーザ制約ファイルの作成

まず,ucfファイルを作ります.Project Navigatorのメニューから「Project」→「New Source」を選択すると,New Source Wizardが開きます.

1ページ目(Select Source Type)では,ソース・コードの種類を指定します.ユーザ制約ファイルを作成するので,「Implementation Constraints File」を選択します.File name欄には最上位ブロックに合わせて「top」と入力します.

2ページ目(Summary)で内容を確認して[Finish]ボタ

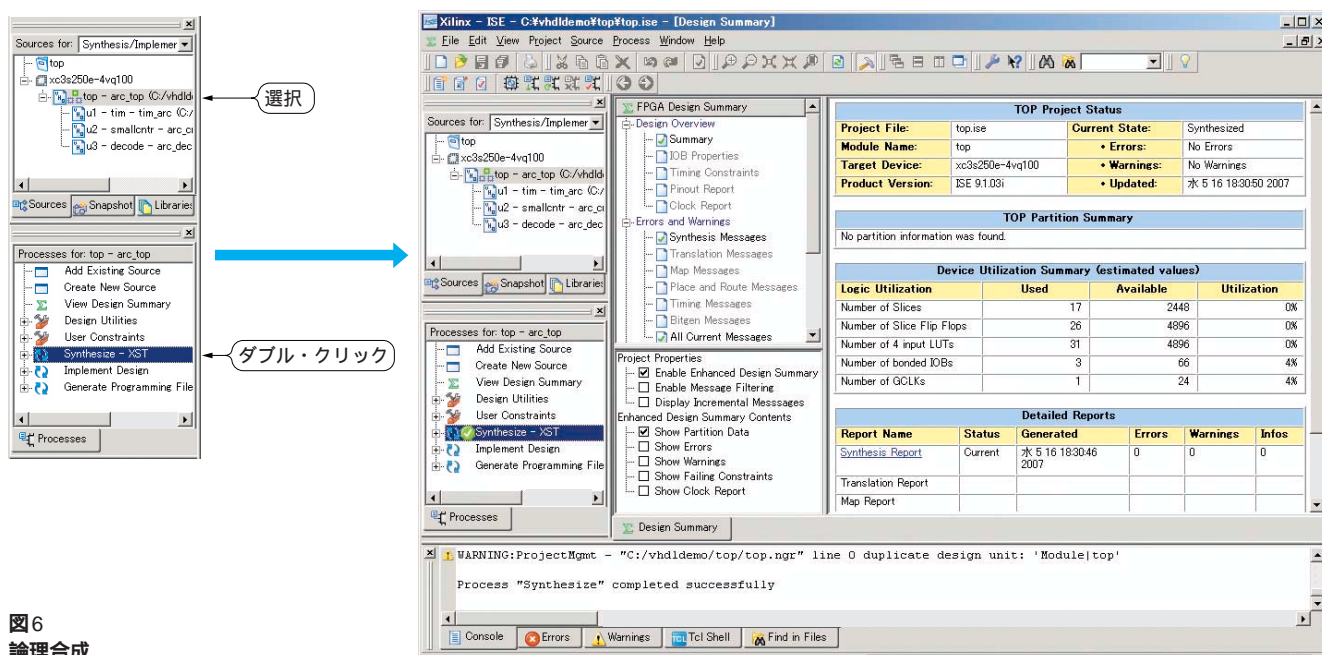
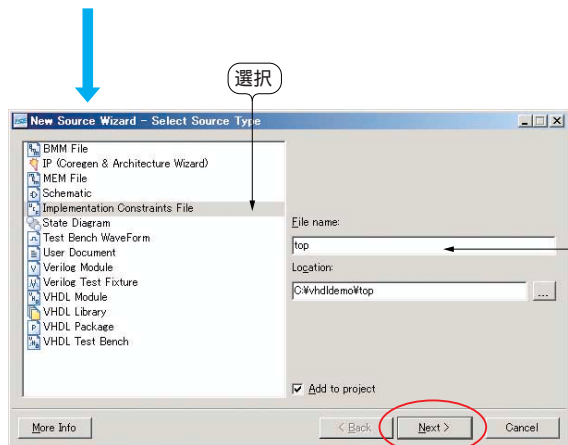
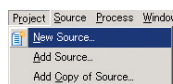



図6
論理合成

ンをクリックします．

Project NavigatorのSources forウィンドウに，top.ucfが追加されます．

(2) PACEによるピン指定

ISEの編集機能を使って，ピン配置の制定を行います．

Project NavigatorのSources forウィンドウでtop.ucfを選択し，Process forウィンドウでUser Constraintsの下にある「Assign Package Pins」をダブル・クリックすると，PACE（編集機能）が起動します．

Design Object Listウィンドウに3本の信号が表形式で表示されているので，Loc欄にピン番号を入力します．こ

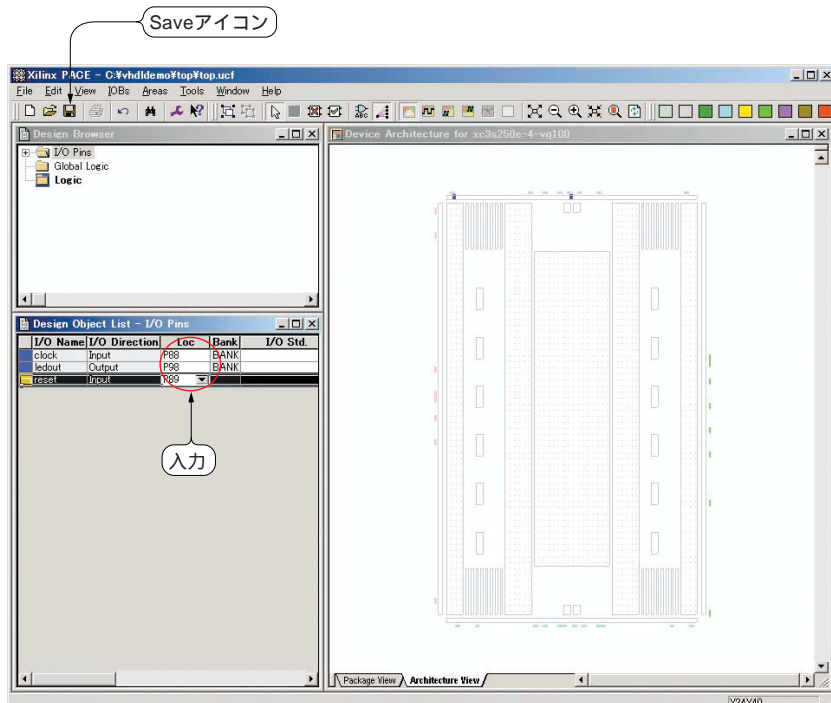
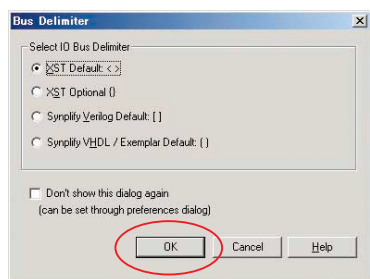
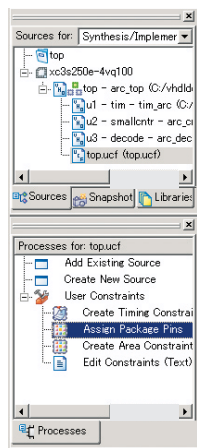
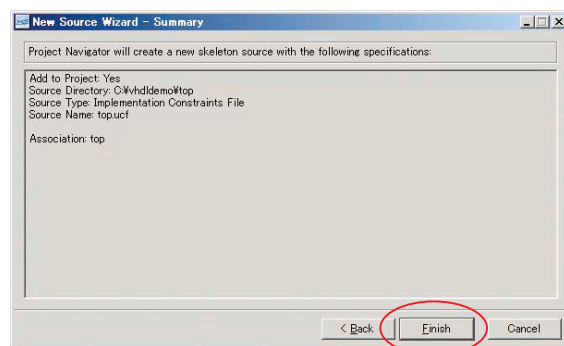



図7
ピン配置の指定

のとき，ピン番号の頭に“P”を付けて入力します．付属FPGA基板の回路に合わせて，clock信号は「P88」，ledout信号は「P98」，reset信号は「P89」とします．入力が終わったらSaveアイコンをクリックして，入力した情報を保存します．Bus Delimiterウィンドウが開いたら，XST Defaultがチェックされていることを確認して［OK］ボタンをクリックします．

これで，ユーザ制約ファイルの中に，ピン指定の情報が書き込まれました．ピン配置の指定が完了したので，PACEウィンドウを閉じます．

● 配置配線を実行する

指定したピン配置に合わせて配置配線を実施します．

Project NavigatorのSources forウィンドウで最上位ブロックのtopを選択し，Process forウィンドウで「Implement Design」をダブル・クリックします(図8)．

配置配線が正常に終了したら，Implement Designの前に緑のチェックが付きます．

● FPGAへの回路情報の書き込み

付属FPGA基板には回路情報を記録するROMは標準では搭載されていません．FPGAに対してJTAGポートから回路情報を書き込みます(図9)．FPGAへの回路情報の書き込みのことを，プログラミングやダウンロード，コンフィグレーションと言います．

(1) プログラム用ファイルの作成

まず，プログラム用ファイルを作成します．FPGAに対して直接書き込む時に使用するのは，拡張子がbitのファイルです．

Project NavigatorのSources forウィンドウで最上位ブロックのtopを選択します．この状態で，Process forウィンドウの「Generate Programming File」を右クリックして表示されるメニューから「Properties...」を選択します．

Process Propertiesウィンドウが開きます．Category欄から「Startup Options」を選択し，FPGA Start-Up Clockを「JTAG Clock」に設定します．ほかはデフォルトのままでかまいません．設定が終わったら［OK］ボタンをクリックします．

Project Navigatorの画面に戻ったら，「Generate Programming File」をダブル・クリックします．

正常に終了すれば，Generate Programming Fileの前に緑のチェックが付き，プロジェクト・フォルダ内にtop.bitファイルが生成されます．

(2) プログラミングの準備

FPGAプログラミング用のケーブル(Xilinx社の「Platform Cable USB」など)でパソコンと付属FPGA基板を接続し，付属FPGA基板に電源を供給します．

図8
配置配線

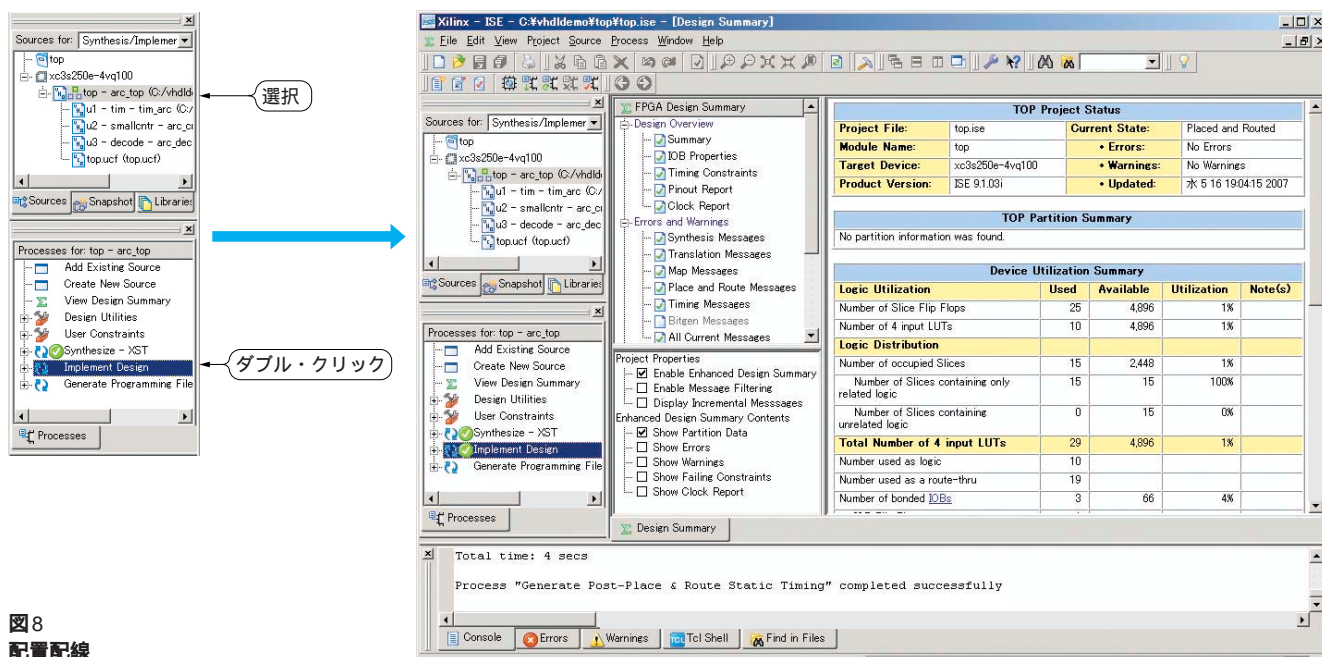
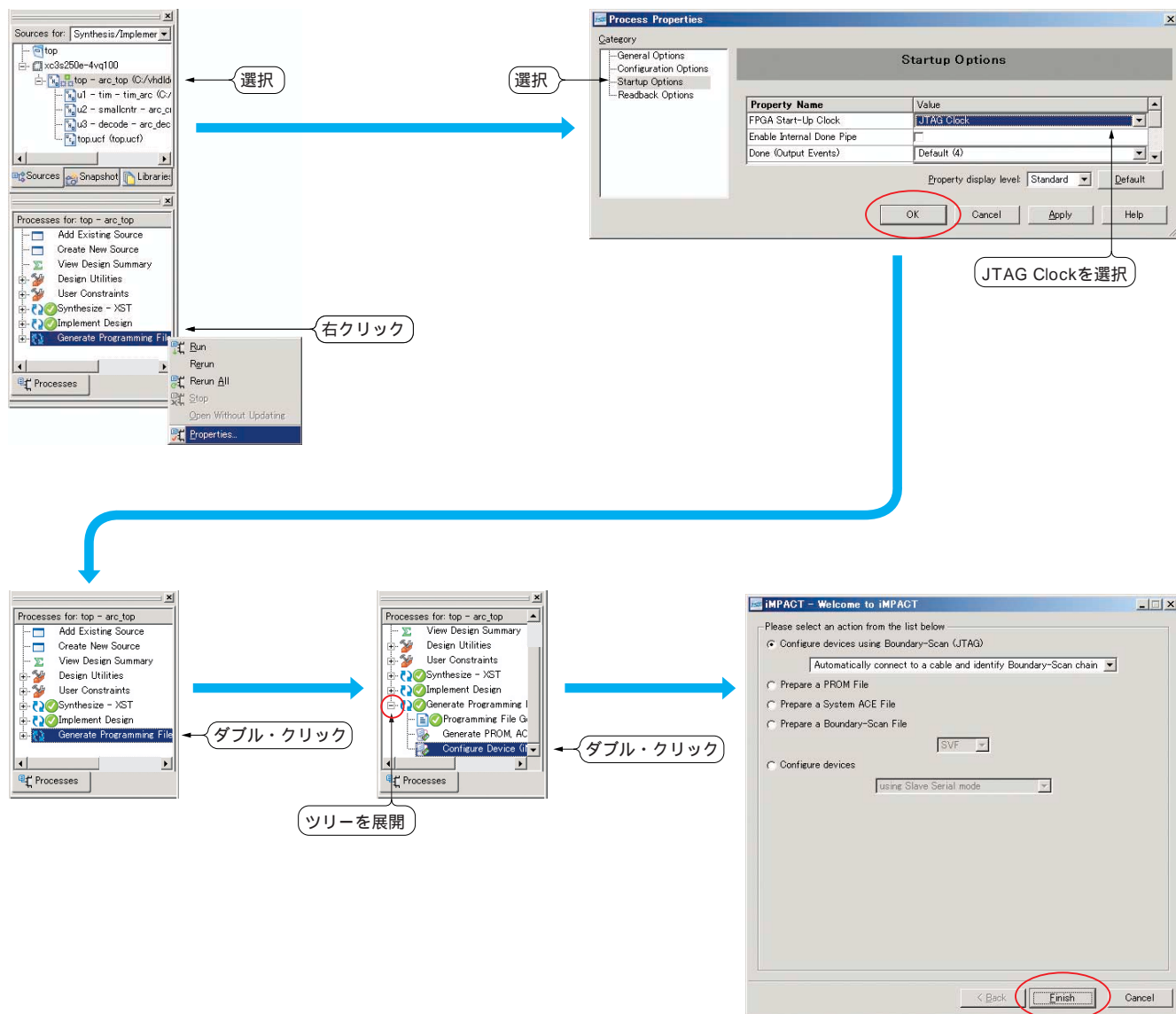



図9　FPGAのプログラミング

(3) プログラミングの実行

Generate Programming Fileの下にある「Configure Device (iMPACT)」をダブル・クリックします．

iMPACTというプログラミング機能のWelcome to iMPACT画面が開きます．「Configure devices using Boundary-Scan (JTAG)」にチェックが入り，「Automatically connect to cable and identify Boundary-Scan chain」が選択されていることを確認して[Finsh]ボタンをクリックします．

画面が切り替わり，Assign New Configuration Fileウィンドウが開いたら，プログラミング用ファイルの「top.bit」を選択して，[Open]ボタンをクリックします．

FPGAのアイコン(xc3s250e)を右クリックして表示されるメニューから「Program」を選択します．Program Propertiesウィンドウが開いたら，デフォルトのまま[OK]ボタンをクリックします．するとFPGAのプログラミングが始まります．

プログラミング中は，Process Dialogウィンドウでプログラミングの進み具合が表示されています．Programming Succeededと表示されればプログラミングは終了です．付属FPGA基板でLEDの点滅が確認できます．

いけざわ・よういち
アヴネット ジャパン(株)

Assign New Configuration File

Look in: C:/vhdldemo/top/

..
_ngo
_xmsgs
xst
top.bit ← 選択

File name: top.bit　Open
File type: All Design Files (*.bit *.rbt *.nky *.isc *.bsd)　Cancel
Cancel All　Bypass

None
Enable Programming of SPI Flash Device Attached to this FPGA
Enable Programming of BPI Flash Device Attached to this FPGA

TDI　XILINX　xc3s250 top.bit　TDO　← 右クリック

Program...
Verify
Get Device ID
Get Device Signature/Usercode
Assign New Configuration File...

Programming Properties

Category
Programming Properties
Advanced PROM Programming Properties
Revision Properties

Programming Properties
General Programming Properties

Verify
General CPLD And PROM Properties
Erase Before Programming　Read Protect
PROM/CoolRunner-II Usercode (8 Hex Digits)
CPLD Specific Properties
Write Protect　Functional Test　On-The-Fly Program
XPLA UES Enter up to 13 characters
PROM Specific Properties
Load FPGA　Parallel Mode　Use D4 for CF
FPGA Device Specific Programming Properties
Pulse PROG　Program Key
Spartan3AN Programming Properties
Data Protect　Data Lockdown

OK　Cancel　Apply　Help

Program Succeeded